CMPE 382 Operating Systems, Spring 2024-2025

Homework 3: Reader-writer problem with password in C Language on Linux

Due Date: May 28, 2025 23:00

Problem Statement:

Implement a solution to the **Reader-Writer problem** (see the textbook and lecture notes) with **password** authentication using threads and semaphores in C programming language on a Linux platform. The program should create two types of threads: **reader** and **writer**. Each thread must register itself by storing a password in a table during the creation of it and be verified with its unique password stored in the table to access the database (a global variable **BUFFER**). There can be a minimum of 1 and a maximum of 9 readers/writers. So, any combination of readers and writers is possible, such as (1,1), (1,2)… (1,9), (2,1) … (9,1), …, or (5,5).

The writers can write a random number in the range 0-9999 to BUFFER, each time it has an access to it. Let each writer/reader sleep 1 second before consecutive writing/reading.

The password table is filled with hash values of the threads in the main process. Each reader/writer registers itself by inserting its hash value as password into the table. The shared resource (**BUFFER**) should only be accessible after being checked for the unique password.

The following code snippet can be used to obtain a hash value for a thread; you may also create your own function:

```
unsigned long hash_pthread_t(pthread_t thread_id) {
  return (unsigned long)(uintptr_t)thread_id;
}
```

)

Furthermore, create equal number of dummy readers and writers that don't register themselves into password table. For example, if there are 2 readers and 3 writers, there will be 2 dummy readers and 3 dummy writers.

Hint: You are free to make any assumption, as long as you document it.

Tasks:

1. Design a program structure that creates reader and writer threads, each requiring a unique password for access.

2. Implement semaphore based synchronization to control access to the password and the shared resource BUFFER.
3. Ensure exclusive access to the password file to prevent race conditions and duplicate passwords.
4. Ensure that readers can access the resource simultaneously with the correct password without interfering with each other.
5. Ensure that writers have exclusive access to the resource with the correct password, preventing simultaneous access by readers or other writers.
6. Test the program with 3 cases with different numbers of readers and writers. Each reader/writer can do 5 operations. The total number of real readers and writers cannot exceed 10.
7. Each time BUFFER is accessed by a real or a dummy reader/writer, the following output is produced in the form of a table. In total, 3 tables are created.

Thread_No Hash_Value Validity(real/dummy) Role(reader/writer) Value read/written

Requirements:

- Utilize POSIX threads (pthread) library for multi-threading.
- Implement appropriate synchronization mechanisms to prevent race conditions.
- Write clear and concise code with proper error handling.
- Include comments to explain the logic and functionality.
- Submit both the source code and the test results.

Resources:

- POSIX Threads Programming: https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html
- Linux Programming Interface by Michael Kerrisk for detailed information on Linux programming concepts and system calls or any similar Linux programming book